D7-R4

Olszewska, J.I.

Published in:
Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management - Volume 2: KEOD

DOI:
10.5220/0008354804350441

Published: 30/09/2019

Document Version
Publisher's PDF, also known as Version of record

D7-R4: Software Development Life-Cycle for Intelligent Vision Systems

J. I. Olszewska
School of Computing and Engineering, University of West Scotland, U.K.

Keywords: Expert Systems, Intelligent Vision Systems, Intelligent Agents, Intelligent Robotics, Autonomous Systems, Machine Vision, Human-machine Cooperation, Cybernetics, Software Engineering, Software-hardware Design, Software Development Life-Cycle.

Abstract: Intelligent Vision Systems (IVS) are omnipresent in our daily life from social media apps to m-health services, from street surveillance cameras to airport e-gates, from drones to companion robots. Hence, IVS encompass any software which has a visual input processed by means of algorithm(s) involving Artificial Intelligence (AI) methods. The design and development of these IVS softwares has become an increasingly complex task, since vision-based systems have evolved into (semi-)autonomous AI systems, usually requiring effective and ethical data processing along with efficient signal processing and real-time hardware/software integration as well as User Experience (UX) and (cyber)security features. Consequently, IVS system development necessitates an adapted software development life-cycle (SDLC) addressing these multi-domain needs, whilst being developer friendly. Hence, we propose in this paper a new SDLC we called D7-R4 which allows developers to produce quality, new-generation IVS to be deployed in real-time and in real-world, unstructured environments.

1 INTRODUCTION

Intelligent vision systems (IVS) provide information resulting from the processing of visual inputs such as still images, online databases, or live video streams captured by camera(s) (Forsyth and Ponce, 2012). Nowadays, IVS are present everywhere in our Society, ranging from autonomous vehicles (Winfield, 2012) to assisted-living devices (Kaaz et al., 2017), from rescue operations (Olszewska, 2017) to video surveillance (Bhat and Olszewska, 2014) like illustrated in Fig. 1.

Hence, the new-generation of IVS allows systems to get higher autonomy as well as further levels of automated reasoning based on visual input and involves softwares integrating AI-based algorithms and/or Application Programming Interfaces (APIs) (Warrier, 2018). Such IVS softwares need to be not only dependable (Meyer, 2006), but also explainable (Samek et al., 2017) and interoperable (Ciccozzi et al., 2017), leading to new challenges for software developers (Olszewska, 2019).

In the past, most of the IVS softwares were developed as research projects outside the framework of any particular Software Development Life-Cycle (SLDC) (Rezazadegan et al., 2017).

More recently, the development of IVS softwares

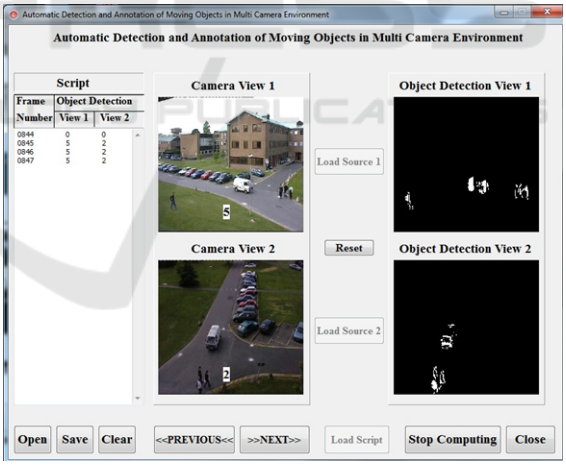


Figure 1: Snapshot of an Intelligent Vision System (IVS) software (Bhat and Olszewska, 2014).

has followed plan-driven methodologies (Sommerville, 2015) or Agile approach (Abrahamsson et al., 2003). Hence, the use of SDLCs like Waterfall (Royce, 1970) assists in a systematic planning of project tasks, but does not incorporate actions such as the adaptation or tuning of a particular AI algorithm. The V-model mainly consists in the production of the requirement documentation and the integration of relevant software units as well as on their associate, thorough, level-by-level testing, rather than on any re-

Olszewska, J.
D7-R4: Software Development Life-Cycle for Intelligent Vision Systems.
DOI: 10.5220/0008354804350441
In *Proceedings of the 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pages 435-441
ISBN: 978-989-758-382-7

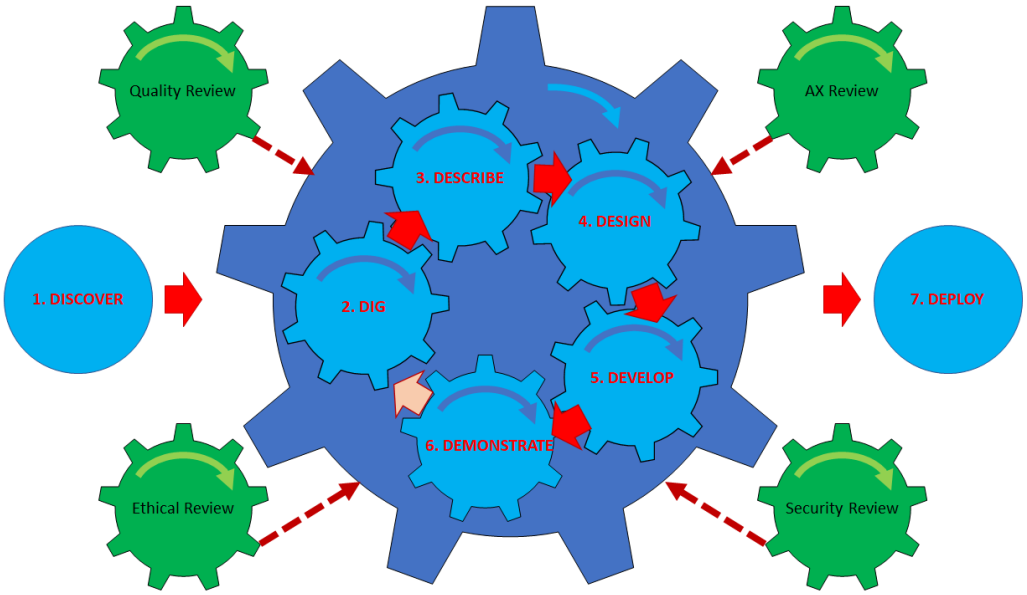
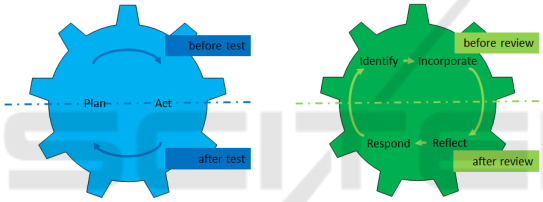


Figure 2: Overview of the D7-R4 Software Development Life Cycle.



(a) (b)

Figure 3: Mechanism details for the following processes within the D7-R4 SDLC: (a) Test iterative process; (b) Review iterative process.

quirement or solution refinement. The prototyping model allows to iteratively refine the requirements, whereas it does not present a mechanism to incorporate user's feedback, e.g. during the User Interface (UI) design. The spiral model (Boehm, 1986) displays an iterative approach to develop software version(s), whilst necessitates the entire product to be produced before getting any user's feedback about it.

On the other hand, the Agile-based Rapid Application Development Model (RAD) (Martin, 1991) has been quite popular to develop effective and user-friendly intelligent systems in short-time frameworks, but the methodology does not permit an in-depth analysis of the data, e.g. to design/manage visual inputs and to consider their ethical implications. The Dynamic System Development Method (DSDM) enables to iteratively determine the software functional design and its development, while it involves a business study not always available in case of new research softwares. Extreme Programming (XP) (Beck, 1999) is an effective methodology to elucidate the requirements and to develop the software by continuously incorporating the user's stories and feedbacks. However, this approach must deal with a client and implies extensive reviewing by this client/user, rather than the refinements of AI algorithms/methods by the developer. The Scrum approach manages the project time in an efficient way by organizing sprints of two to four weeks. During each sprint, the developers create a product increment. Though, the full Scrum methodology is intrinsically not adapted for lone developers such as individual researchers.

As none of the existing SLDCs covers the increasingly complex needs and the multi-disciplinary specifications of the new generation of IVS softwares, we propose a new SDLC called D7-R4 in order to develop such IVS softwares.

This SDLC (Fig. 2) is an Agile, iterative, and test-driven development (TDD) approach (Fig. 3), which could be used by either a lone developer, by a group of software developers, or by an artificial agent-human team.

The D7-R4 software engineering model provides a systematic and ethical way to develop IVS softwares. Indeed, IVS are alloying together information systems, computer vision systems, expert systems, and embedded systems, as well as blending autonomous systems (AS) (Fiorini et al., 2017), user experience (UX) (Ferre and Medinilla, 2007), and cybersecurity (Peruma and Krutz, 2018) capabilities, requiring among others actions such as image/video signal processing (Zendel et al., 2017), visual database management/processing (Ammirato

et al., 2017), AI/Machine Learning algorithms/API implementation/integration (Warrier, 2018), UI design (Dix et al., 2004); all these actions implying explainable AI and ethical considerations.

Thus, the contributions of this paper are manifold. On one hand, we present a new SDLC called D7-R4 to address the new challenges raised by the use of technologies relying on Artificial Intelligence and Machine Vision. On the other hand, we provide a new SLDC which (i) includes an ethical review in the process itself, (ii) is cybersecurity-driven, (iii) inherently incorporates quality in its process, (iv) allows interoperability. Moreover, we introduce the *agent-friendly* concept along the Agent Experience (AX), extending the user-friendly and UX notions, respectively.

The paper is structured as follows. Section 2 describes the new IVS software development life-cycle called D7-R4. Experiments are described in Section 3, while conclusions are drawn up in Section 4.

2 PROPOSED SDLC

2.1 D7-R4 SDLC Overview

The D7-R4 SDLC provides designers and developers with a framework to build reliable, ethical, quality, and secure IVS. It consists in seven stages and four reviews described in Section 2.2 and Section 2.3, respectively. The first stage (i.e. *discover*) is the input of the project or kick-off point, while the final stage (i.e. *deploy*) is the final product consisting of all the project outputs which have been successively produced in the different runs of the five core stages, i.e. *dig*, *describe*, *design*, *develop*, and *demonstrate* (Fig. 2). These five core components could be iteratively repeated in sprints, and each of these five core steps could be carried out in loops; the number of the sprints and loops being variable and adjustable considering the requirements, size, and progress of the project/product. Indeed, D7-R4 is a flexible process, and its components are modular, i.e. they could be followed entirely or partially, depending of the project scope and/or product development needs. It is worth noting that IVS projects could start from scratch or build upon a previous project and/or reuse existing components such as algorithms, libraries, API, etc. Figures 3(a)-3(b) show the different mechanism applied to the iterative processes of testing and reviewing, respectively, as described in the next section.

2.2 D7-R4 SDLC Stages

2.2.1 Discover Stage

This phase is the first stage of the SDLC and consists in a brief context study, as well as in the quick capture of the project scope and aims.

2.2.2 Dig Stage

This step focuses on the research and feasibility study. This could include actions such as the project domain study (e.g. identification of the specific technical challenges, identification/selection of the pertinent state-of-the-art methods, existing tools, appropriate research methods), the project planning (in terms of time and resources), the risk analysis, a business study (in terms of foreseen costs/available budget/potential market), and the quality planning. As the initial conditions of the project (e.g., on one hand, the allocated budget, the market opportunities, and on the other hand, the available tools, the state-of-the art methods, etc.) could evolve during the project, even within short period of time, this stage could be subject to further iterations, if need be.

2.2.3 Describe Stage

This phase consists in the requirement gathering, e.g. the expression of the IVS software functionalities (by means of the MoSCoW analysis), the capture of the business/system/user requirements as well as the database requirements (persistence) and the Human-Computer Interactions (HCI)/User Interface (UI)/ User Experience (UX) needs. This task also includes a reflection on the associated professional, social, environmental and legal (PSEL) considerations as well as on the safety and quality assurance and relevant norms and standards (Olszewska et al., 2018). It is worth noting that based on the user/client/developer feedback during the design and development process, the requirements may be amended.

2.2.4 Design Stage

This stage contains three components, namely, the software design, the database design, and the algorithm design. The software design itself involves the software analysis using e.g. the Unified Modeling Language (UML), the software architecture/data flow modeling e.g. by means of Data Flow Diagrams (DFDs), including aspects of software modularity and extendibility, and the choice of the software language(s), Integrated Development Environment (IDE), design pattern(s), including reflections

on software portability, accessibility, and adaptability. The database design implies the visual input data analysis, including the determination of the data ground truth, training/testing datasets, etc. The algorithm design is an iterative process in itself and consists of the establishment of the algorithm specifications, the algorithm formalization, the algorithm implementation and debugging, as well as the algorithm performance review based on metrics to assess, in particular, its accuracy, computational efficiency, complexity, and reliability (Olszewska, 2019). After that, the algorithm could undergo the training and testing processes, which can also been carried out iteratively.

2.2.5 Develop Stage

This stage is dedicated to the software development and integration. The software development includes the software implementation along of the integration of previously developed algorithm. This phase is coupled with software unit testing (using both white box and black box testing techniques, test plans, code inspection, etc.). The software integration consists in the integration of all the software units and of the hardware/software integration, if appropriate, as well as of the integration testing and system testing. System performance testing and capability testing could also been performed at this stage, in an iterative way.

2.2.6 Demonstrate Stage

This step aims to assess the UX of the IVS. It can incorporate tests such as further robustness testing, user acceptance testing, interactivity testing, software usability, learnability as well as interoperability testing.

2.2.7 Deploy Stage

This is the final stage which ends in the IVS software release. This action could require an internal approval by the developer/manager/client and an external approval by relevant professional bodies, if appropriate.

2.3 D7-R4 SDLC Reviews

2.3.1 Quality Review

The quality management (Futong and Tingting, 2013) review could be performed at different stages in the described process, e.g. at stages 2, 4, and 6 (Fig. 2). It is worth noting that, in this approach, the project/product boundaries are blurred to accommodate situations where products such as autonomous systems can be required to do by themselves or by teaming with human(s) a series of stages to accomplish a project aiming their own reconfiguration, in order e.g. to fix their softwares' errors or to change requirements, modify plans, etc. to adapt to new situations.

2.3.2 AX Review

This review aims to provide an up-to-standard Agent Experience (AX) (Olszewska and Allison, 2018). It includes usability tests, while it extends User Experience (UX) concept, since the review is not only user-focused but tends to be 'agent-friendly' for any intelligent agent, i.e. a human agent such as user/designer/expert, or an artificial agent such as expert systems, robots, etc. getting to interact with that interoperable system. The AX review could be carried out at stages 3, 4, 5, and 6 (Fig. 2).

2.3.3 Ethical Review

Integrating ethical considerations in the cycle is of prime importance for intelligent systems (Wallach and Allen, 2009) such as IVS. The review could address various questions, as follows: is the database compliant with the General Data Protection Regulation (GDPR), where the data does come from, how they are kept, is there a bias in collecting the data (Fang et al., 2013), is there a bias in the algorithm when processing them, what are the true positive (TP) rate and the true negative (TN) rate of the training dataset, etc. The review could also reflect on aspects such as the safety of the final behaviour of the system, the evolution of the IVS system behaviour over time, its dual applications and possible misuses, etc. This review could be done at stages 2, 3, 4, and 6 (Fig. 2).

2.3.4 Security Review

The process allows a system security review (Peruma and Krutz, 2018) which should be adapted to the specific project requirements and run periodically. In particular, this review should identify any transmitted and/or stored data, especially if containing sensitive information and ensure it is encrypted accordingly. This review could be integrated at stages 2, 3, 4, and 6 (Fig. 2).

3 EXPERIMENTS

Experiments have consisted in collecting primary data from the available documentation of 84 supervised/led projects aiming to develop intelligent systems, in order to identify the main advantages and

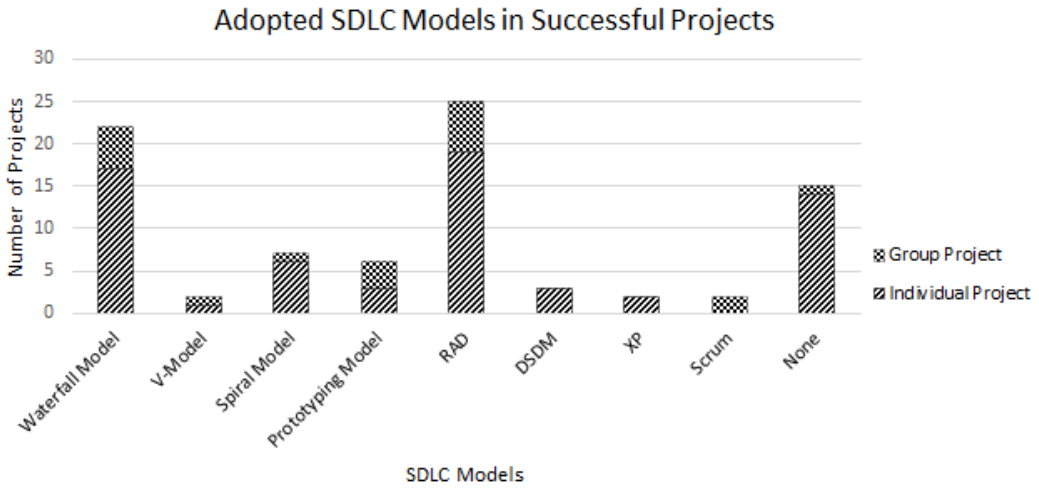


Figure 4: Sample results of the adopted SDLC models in successful AI-based projects.

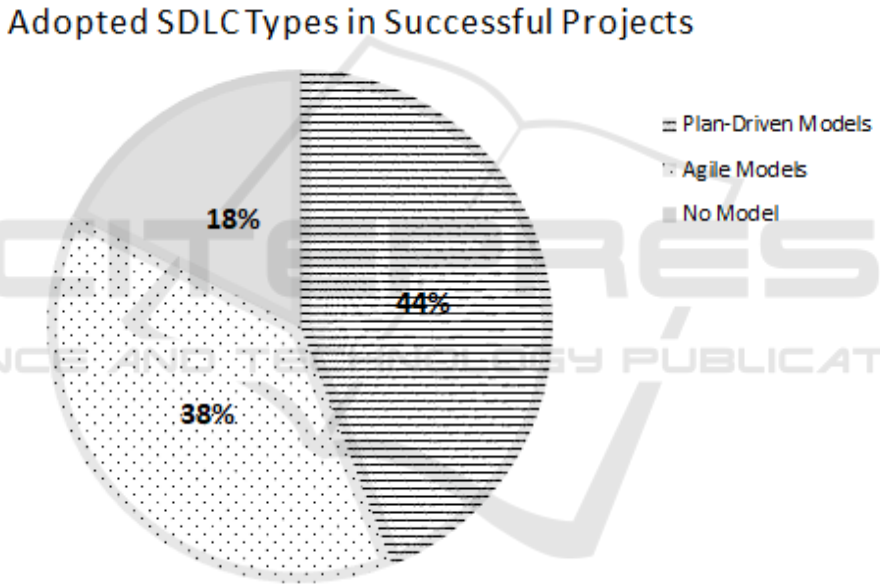


Figure 5: Sample results of the adopted SDLC types in successful AI-based projects.

drawbacks of the existing SDLC models as reported in Section 1. The typical duration of these projects was between 2 to 12 months. Projects were carried over the five past years and were all tested successfully as per standards (Beller et al., 2018a), (Beller et al., 2018b).

Results displayed in Fig. 4 show that the most popular SDLCs were the Waterfall model and the RAD model, one due to its clear sequence of tasks and the other one for the ease of feedback incorporation and the possibility of iterative refinement. The further analysis of the software documentations shows that, in the past, one developer over five did not find an adequate SDLC among the existing ones to develop an IVS system (Fig. 5), whereas, recently, a pilot group of 5 software developers has unanimously adopted the new SDLC named D7-R4, which has thus been tested on all their 5 projects; the IVS outputs being very successfully delivered as per internal and external, relevant assessments.

4 CONCLUSIONS

In order to develop intelligent vision systems (IVS) with an increasing degree of sophistication, we propose a new Software Development Life-Cycle (SDLC) we called D7-R4. The presented SDLC is

an Agile, iterative approach which is inherently test driven and which consists of seven stages, namely, *discover*, *dig*, *describe*, *design*, *develop*, *demonstrate*, and *deploy*, as well as four reviews from different perspective such as quality, user/agent experience, ethics, and security. This D7-R4 methodology has been successfully applied to develop recent as well as current IVS projects and shows promising results for IVS ranging from computer-vision systems to vision agents.

REFERENCES

Abrahamsson, P., Warsta, J., Siponen, M. T., and Ronkainen, J. (2003). New directions on Agile methods: A comparative analysis. In *Proceedings of the IEEE International Conference on Software Engineering (ICSE)*, pages 244–254.

Ammirato, P., Poirson, P., Park, E., Kosecka, J., and Berg, A. C. (2017). A dataset for developing and benchmarking active vision. In *Proceedings of IEEE Conference on Robotics and Automation (ICRA)*, pages 1378–1385.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change*. Addison-Wesley.

Beller, M., Georgios, G., Panichella, A., Proksch, S., Amann, S., and Zaidman, A. (2018a). Developer Testing in The IDE: Patterns, Beliefs, And Behavior. *IEEE Transactions on Software Engineering*.

Beller, M., Spruit, N., Spinellis, D., and Zaidman, A. (2018b). On the dichotomy of debugging behavior among programmers. In *Proceedings of IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 572–583.

Bhat, M. and Olszewska, J. I. (2014). DALES: Automated tool for detection, annotation, labelling and segmentation of multiple objects in multi-camera video streams. In *Proceedings of the ACL International Conference on Computational Linguistics Workshop (COLING)*.

Boehm, B. W. (1986). A spiral model of software development and enhancement. In *ACM SIGSOFT Engineering Notes*, pages 14–24.

Ciccozzi, F., Ruscio, D. D., Malavolta, I., Pelliccione, P., and Tumova, J. (2017). Engineering the software of robotic systems. In *Proceedings of the IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 507–508.

Dix, A., Finlay, J., Abowd, G. D., and Beale, R. (2004). *Human Computer Interaction*. Pearson, 3rd edition.

Fang, C., Xu, Y., and Rockmore, D. N. (2013). Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias. In *Proceedings of IEEE Conference on Computer Vision (ICCV)*, pages 1657–1664.

Ferre, X. and Medinilla, N. (2007). How a human-centered approach impacts software development. In *Proceedings of the International Conference on Human-Computer Interaction (HCI)*, pages I.68–I.77.

Fiorini, S. R., Bermejo-Alonso, J., Goncalves, P., de Freitas, E. P., Alarcos, A. O., Olszewska, J. I., Prestes, E., Schlenoff, C., Ragavan, S. V., Redfield, S., Spencer, B., and Li, H. (2017). A suite of ontologies for robotics and automation. *IEEE Robotics and Automation Magazine*, 24(1):8–11.

Forsyth, D. A. and Ponce, J. (2012). *Computer Vision: A Modern Approach*. Pearson.

Futong, H. and Tingting, S. (2013). Software project metrics and quality management. In *Proceedings of IEEE Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 615–618.

Kaaz, K. J., Hoffer, A., Saeidi, M., Sarma, A., and Bobba, R. B. (2017). Understanding user perceptions of privacy, and configuration challenges in home automation. In *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 297–301.

Martin, J. (1991). *Rapid Application Development*. Macmillan.

Meyer, B. (2006). *Dependable Software*, pages 1–33. Springer.

Olszewska, J. I. (2017). Clock-model-assisted agent's spatial navigation. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, pages 687–692.

Olszewska, J. I. (2019). Designing transparent and autonomous intelligent vision systems. In *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART)*, pages 850–856.

Olszewska, J. I. and Allison, I. K. (2018). ODYSSEY: Software Development Life Cycle Ontology. In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development (KEOD)*, pages 303–311.

Olszewska, J. I., Houghtaling, M., Goncalves, P., Haidegger, T., Fabiano, N., Carbonera, J. L., Fiorini, S. R., and Prestes, E. (2018). Robotic ontological standard development life cycle. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6.

Peruma, A. and Krutz, D. (2018). Understanding the relationship between quality and security: A large-scale analysis of android applications. In *Proceedings of IEEE/ACM International Workshop on Security Awareness from Design to Deployment*, pages 19–25.

Rezazadegan, F., Shirazi, S., Upcrofit, B., and Milford, M. (2017). Action recognition: From static datasets to moving robots. In *Proceedings of IEEE Conference on Robotics and Automation (ICRA)*, pages 3185–3191.

Royce, W. W. (1970). Managing the development of large software systems. In *Proceedings of IEEE Conference of Western Electronic Show and Convention*, pages 205–210.

Samek, W., Wiegand, T., and Muller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ITU*

Journal: ICT Discoveries - Special Issue 1 - The Impact of Artificial Intelligence (AI) on Communication Networks and Services, 1:1–10.

Sommerville, I. (2015). *Software Engineering*. Pearson, 10th edition.

Wallach, W. and Allen, C. (2009). *Moral Machines: Teaching Robots Right from Wrong*. Oxford University Press.

Warrier, G. (2018). Breaking The Common Myths Around Artificial Intelligence. In *DDD Scotland*.

Winfield, A. (2012). *Robotics: A Very Short Introduction*. Oxford University Press.

Zendel, O., Honauer, K., Murschitz, M., Humenberger, M., and Dominguez, G. F. (2017). Analyzing computer vision data - the good, the bad and the ugly. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6670–6680.